SAND-2002-3276
Unlimited Release
Printed October 2002

An Evaluation of the Convergence Properties of a Parallel Genetic Programming Method

Dianne C. Barton
Critical Infrastructure Surety Department

R. J. Pryor
Computational Biology and Evolutionary Computing Department

Sandia National Laboratories
P.O. Box 5800
Albuquerque, NM 87185-0321

Abstract

Genetic programming is a technique that produces as output the source code of another computer program. The program is evolved with rules of natural selection that seek to find the best solution to a particular problem. This report presents the results of tests run on eight different GP method parameters and shows how variations in the value of these parameters affects the time taken to converge on the correct solution. The problem used to test the method is the MAX problem which proved to be a simple, straight forward, and easy approach for evaluating GP efficiency. Testing the method on the MAX problem can help to develop an optimum search for problems with unknown solutions. This paper presents the results of this study.

Intentionally Left Blank

Contents

Figures

Tables

An Evaluation of the Convergence Properties of a Parallel Genetic Programming Method

Introduction

Genetic programming (GP) described in Koza (1992) is an approach to automatic programming in which computers are used to write programs. GP uses an evolutionary search technique to find a highly fit computer program in the space of possible computer programs for solving a particular problem. It is a useful approach because it may find novel solutions that might be overlooked in human designed solutions. This is because computers opportunistically attempt to solve the problem without the constraint of human preconceptions about an appropriate solution path. The technique has been used by Pryor (1998) to develop behavior for tracking robots, by Barton (2002) to develop multiple pricing strategies for an agent based economic model, and by Pryor and Barton (2002) to find manuevering behaviors for an unmanned aerial vehicle glider (UAV).

An important aspect of designing a successful GP method is in developing optimum settings for the GP search parameters. This is a difficult problem when the solution space is unknown. Therefore, we conducted tests of a GP method on the MAX problem and measured the time to converge on the correct solution as the GP parameters changed. Testing the method on the MAX problem can help to develop an optimum search for problems with unknown solutions. It is simple, straight forward, and easy to conduct. This paper presents the results of this study.

Genetic Programming

GP is a technique that produces as output the source code of another computer program. GP uses the ideas of biological evolution to solve a complex problem by employing the principles of natural selection, where the most effective programs survive and compete or cross-breed with other programs, to continually approach the best solution to a particular problem.

GP starts with an initial population of randomly generated computer programs composed of functions and terminals appropriate to the problem domain. The size of the population varies based on the problem but thousands of individuals or programs are typical. Each program that is generated is evaluated for its ability to solve the prescribed task and is assigned a numerical score or fitness for how well it has achieved this goal. Some programs will be very effective at accomplishing the prescribed task and some will not. The higher the fitness, the better the individual. A new population is created by performing genetic operations including reproduction, crossover and mutation on the individual programs that make up the starting population. The old population is then discarded and the process is repeated using the new population. One iteration of this loop is referred to as a generation.

The following sections describe in detail the tree representation, the genetic operators, and the solution procedures for the GP method that was used in this study. The input variables that were tested and the input parameters that were used as part of this study are also described in the appropriate sections.

Program Representation

Programs can be graphically depicted as a rooted point labeled tree with ordered branches. Program "(a * b) – c + (d * e)" would be represented as the tree shown in Figure 1. The basic building block of a tree is called a *node*, with all nodes in the tree having the same fixed structure. A node can either be a function or a terminal. A *function node* performs a mathematical or Boolean operation and generally has branches that point to other nodes. The number of branches depends on the *kind* of function, e.g., add, subtract, multiply. A *terminal node* normally returns a value, does not have any branches, and terminates that section of the tree. The root node is counted as level 0. The tree shown in Figure 1 is three levels deep, nine nodes, four function nodes, and five terminal nodes.

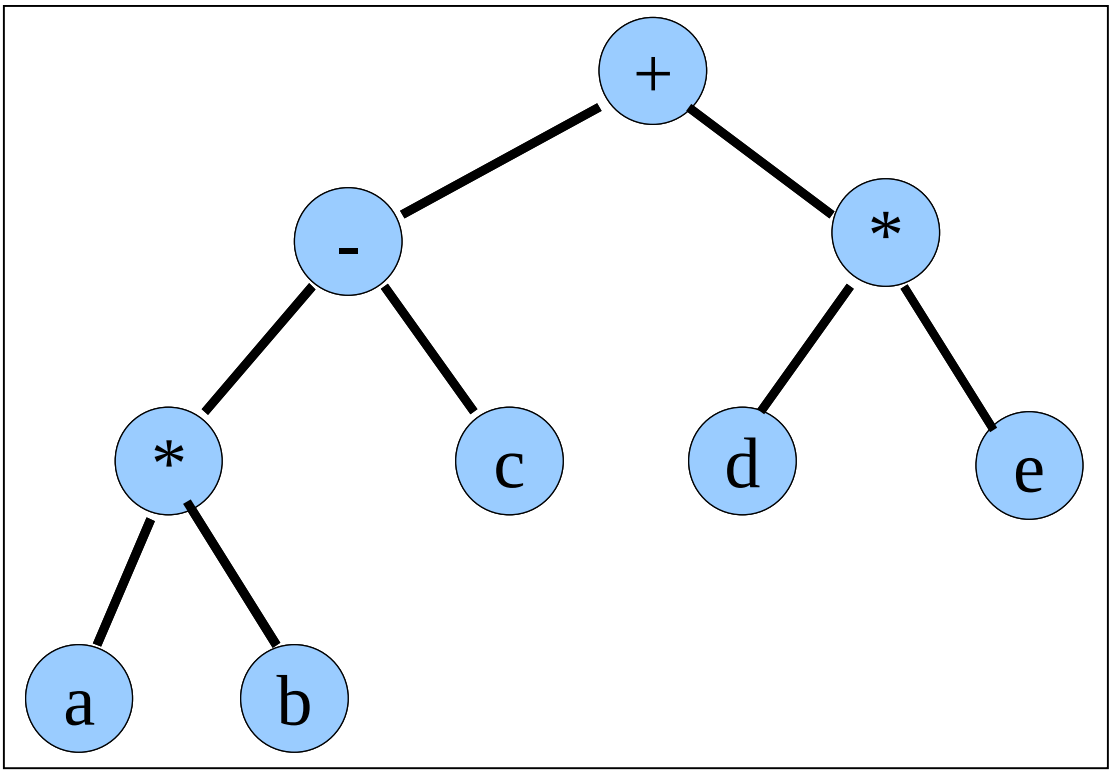

Figure 1 – Parse Tree Representation

Two parameters are used to control the size of a tree, with size determined by the number of levels. The parameter MINTREESIZE specifies the minimum number of levels in a tree, while the parameter MAXTREESIZE specifies the maximum number of levels in a tree. These parameters are used in the following way. If the level number is less than MINTREESIZE, the recursive function that builds the tree will only select from the *function* node kinds, e.g., add,

subtract. This constraint ensures that at least one more level will be added. If the level number is equal to MAXTREESIZE, the recursive function will only select from the *terminal* node kinds. This constraint ensures that no more levels are added to this part of the tree because terminal nodes do not have any branches. For all other level values, a random selection is made.

Genetic Operators

Principles of natural selection are used to create a new generation of individual computer programs from the current population of programs. Three genetic operations are used to create the new generation: reproduction, crossover, and mutation. Each operator works independently of the other operators and the usage of each operator is determined by its assigned probability. The reproduction, crossover, and mutation operators use a selection operation to determine which individuals in the current population will be acted upon.

The population size POPSIZE of each generation remains the same. To create the next generation, a loop over all individual programs in the populations is started. Within this loop, a random number between 0 and 1 is drawn to determine how to replace that individual in the new population. This random number is compared to three probabilities. The probability, PROBMU, determines the percentage of times that the mutation operation is used and the probability, PROBCR, determines the percentage of times that the crossover operation is used. The difference between PROBMU and/or PROBCR and 1.0 is the probability that the reproduction operation is used.

More specifically in our GP method, if the random number chosen at the start of the loop is less than parameter PROBMU, then the individual program is altered by the mutation operation. If the random number is greater than PROBMU but less than PROBCR then the individual program is altered by the crossover operation. If the random number is greater than PROBCR, then the individual is replaced through the reproduction operation. When the required number of new individuals is created, the loop is terminated. Figure 2 is a depiction of the flow of the operation selection procedure. PROBMU and PROBCR are two of the parameters evaluated in this study.

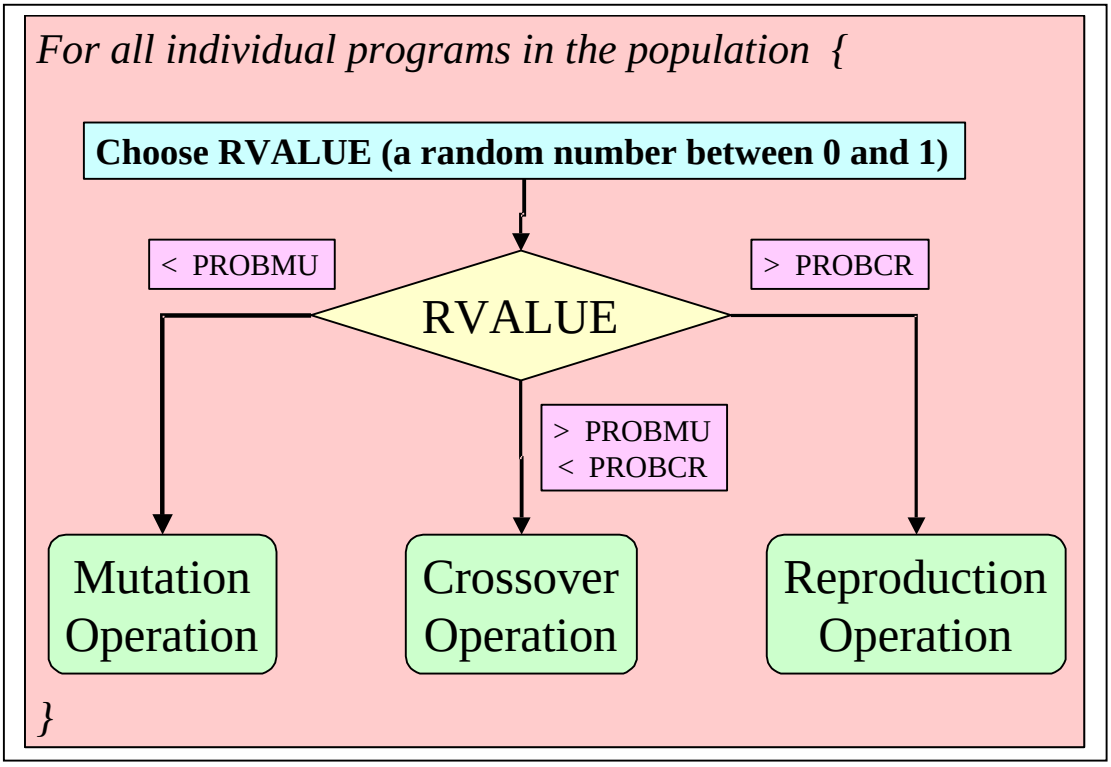

Figure 2 – Flow of genetic operation selection procedure

Selection Operator

The selection operator is used to identify individuals in the current population for reproduction, crossover, or mutation operations. A tournament algorithm is used, which is easy to implement and is relatively fast. The algorithm works in the following way. A SELECTNUM number of individuals are randomly selected from the population. The fitness values of these individuals are compared, and the individual with the highest fitness is the winner of the tournament; that is, the one selected. If two individuals are needed, as in the case of crossover, the tournament is repeated.

SELECTNUM is one of the parameters evaluated in this study. Note that the parameter SELECTNUM affects the distribution of individuals selected, thus increasing the value of SELECTNUM moves this distribution toward more elite individuals in the population. For example, if the value of SELECTNUM is equal to the population size, only the most fit individual will be selected. Reducing the value of SELECTNUM improves diversity by allowing more individuals to take part in forming the next generation.

Reproduction Operator

The reproduction operator uses the selection operator to select a new individual. Thus the operation involves selecting a SELECTNUM number of individuals. The fitness of these individuals is compared and the computer program with the highest fitness is selected for reproduction. This individual program is copied exactly from the current population into the new

population. The reproduction operation does not create anything new in the population of computer programs.

Crossover Operator

The crossover operation creates a new computer program by selecting two parent programs using the selection operator. Next, for each selected individual, a cutpoint is randomly selected among the nodes of its tree. Finally, the new tree is created by removing the cutpoint node and all nodes below it from the first tree and replacing them with the cutpoint node and all nodes below it from the second tree. Figure 3 illustrates this last step. The new tree, which was created by splicing together two trees in the current population, is then placed in the next generation. Unlike reproduction, the crossover operation creates new individuals in the population. By recombining randomly chosen parts of somewhat effective programs, new programs may evolve that are even fitter in solving the problem.

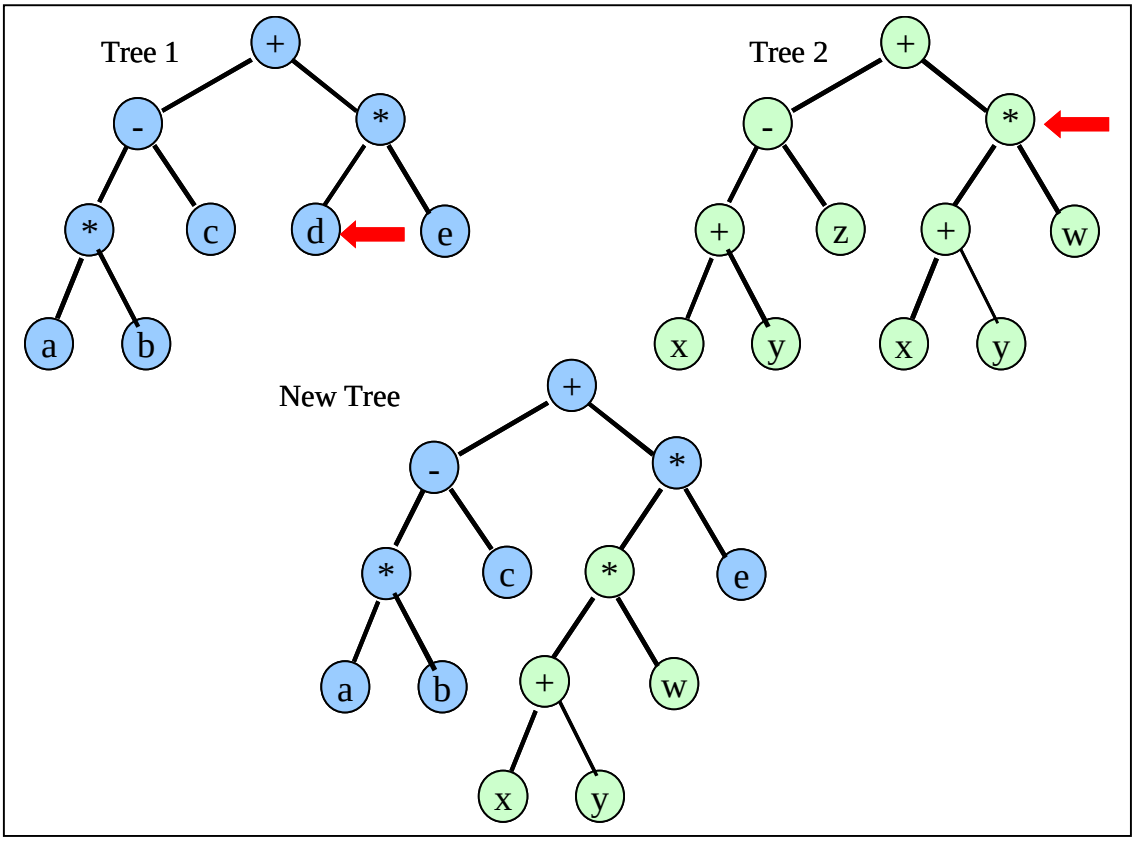

Figure 3. Illustration of the crossover operator – cutpoints for each of the parent trees are shown by the arrows.

Mutation Operator

The mutation operation allows new individuals to be created by randomly changing the values of functions and/or terminals in a selected individual parse tree. Two different mutation styles are tested in this study. In the first type of mutation operation, limb replacement, mutation occurs by choosing a point in the tree at random and generating a new limb at that point to some random

depth not greater than the maximum allowed tree depth, i.e. MAXTREESIZE. This is graphically depicted in Figure 4. In the second type of mutation operation, node swapping, each point in the tree may be altered by a specified probability. The probability that the swap will occur is determined by the probability, PROBNS. Therefore, not all nodes will necessarily be swapped. The node swapping operation is depicted in Figure 5. Terminals are swapped with alternate terminals from the terminal set and functions are swapped with functions with the same number of arguments from the function set. PROBNS is one of the parameters evaluated in this study.

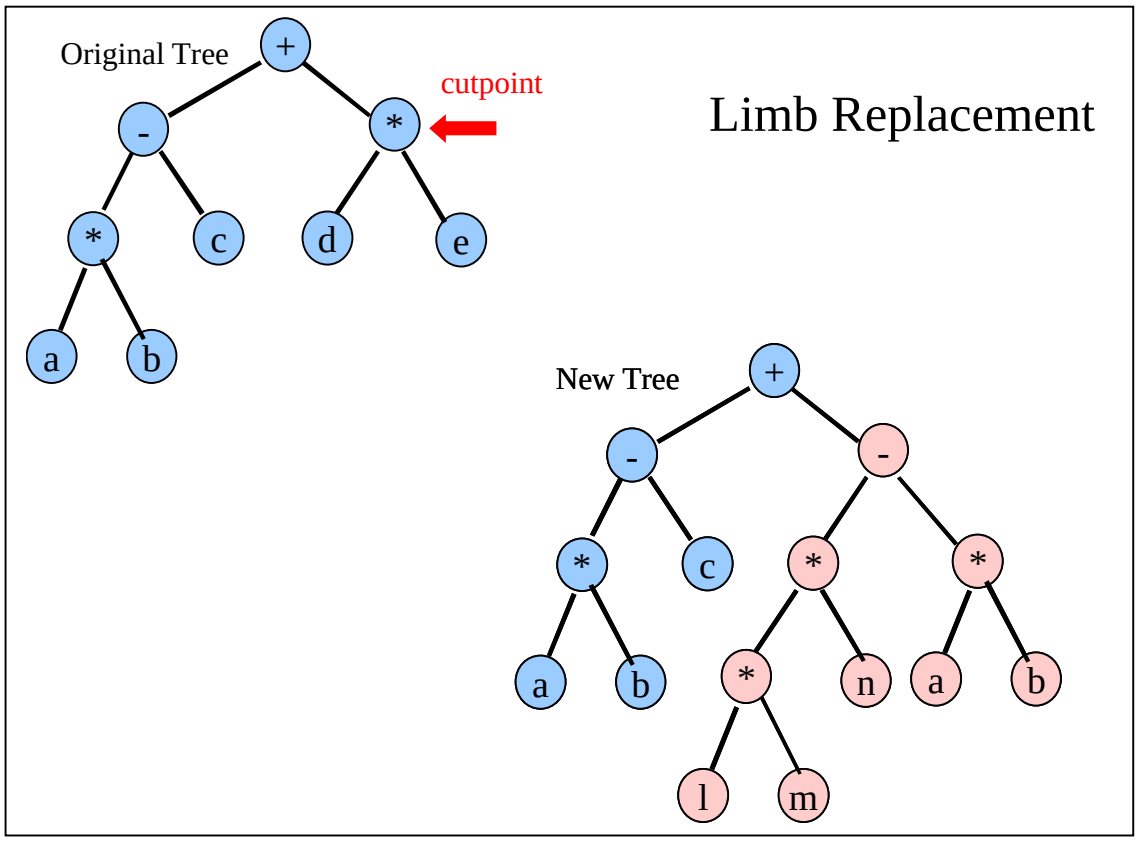

Figure 4 – Illustration of the limb replacement mutation operation – cutpoint is shown by the arrow.

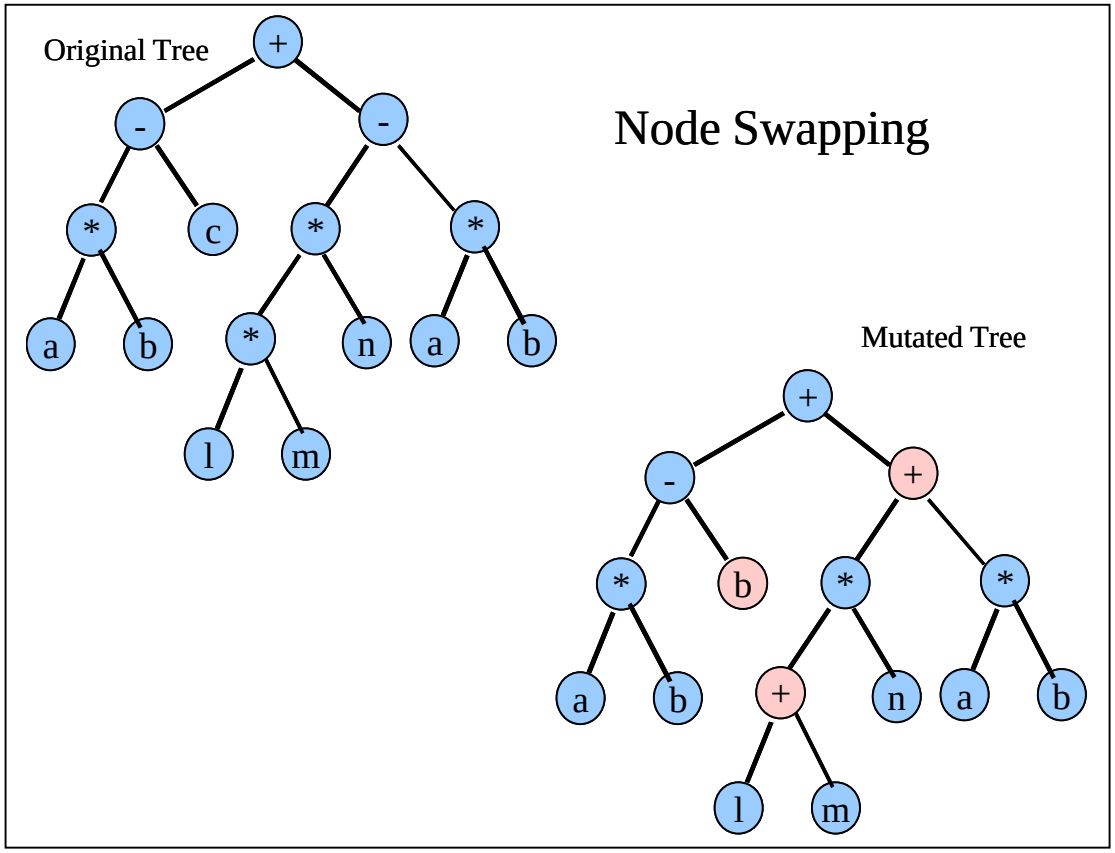

Figure 5 – Illustration of the node swapping mutation operation.

The probability, PROBLB, determines the percentage of times that the limb replacement mutation operation is used versus the percentage of times that the node swapping operation is used. The process is illustrated in Figure 6. First a tree is selected from the current population using the selection operator. Next, a random number is chosen between 0 and 1.0. If this random number is less than PROBLB, then limb mutation is performed on the selected tree; if it is greater, then node swapping mutation is performed. If the node swapping mutation operation is selected, then each node is swapped with the probability PROBND. PROBLB and PROBND are two of the parameters evaluated in this study.

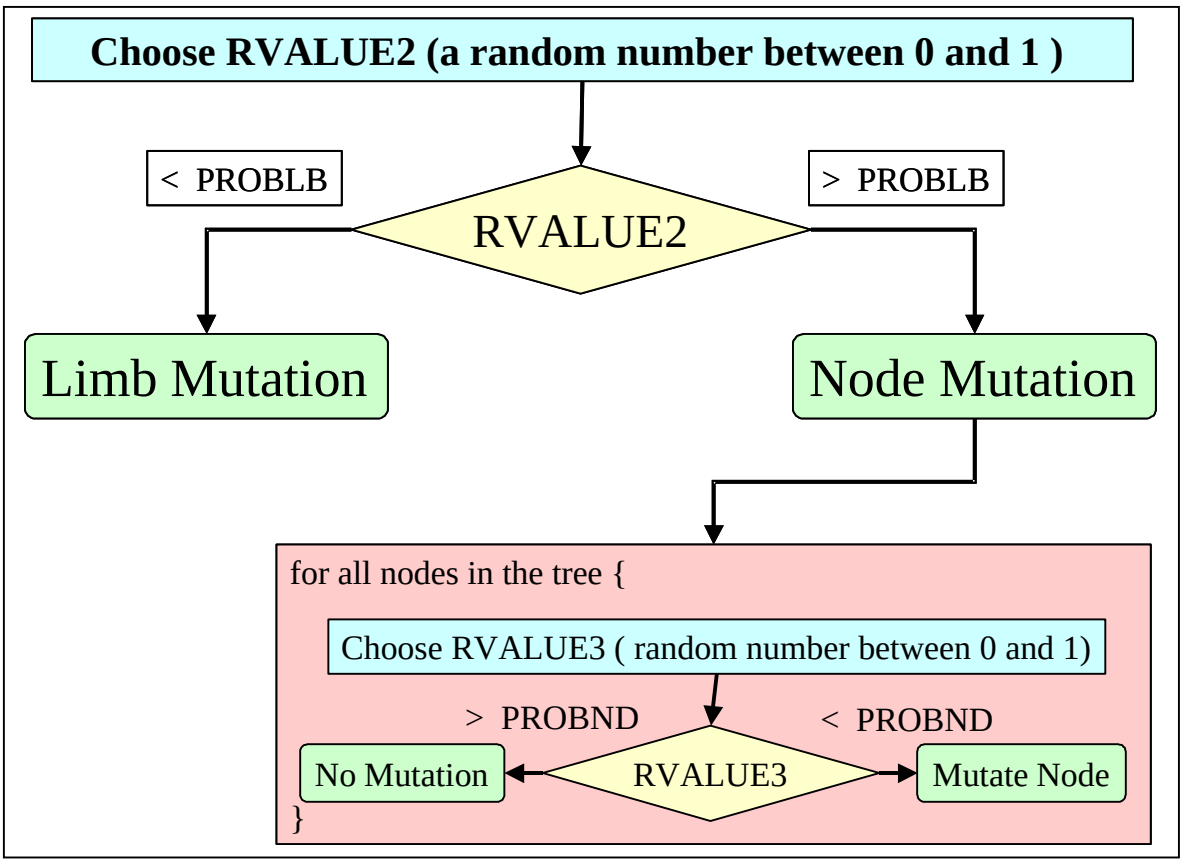

Figure 6 – Flow of mutation operation selection procedure.

Solution Procedure

Figure 7 illustrates the basic solution procedure. The production of the initial generation of programs is done by randomly generating a rooted, point labeled parse tree from a set of functions and terminals. The fitness of each individual in the population is then determined, followed by the initiation of a loop over generations. Within the generation loop, the next generation is created and the fitness of each individual within that generation is calculated. A test is then made to determine whether any individual meets the convergence criterion. If an individual is found, the loop terminates and the calculation ends. If no individual is found, the calculation continues.

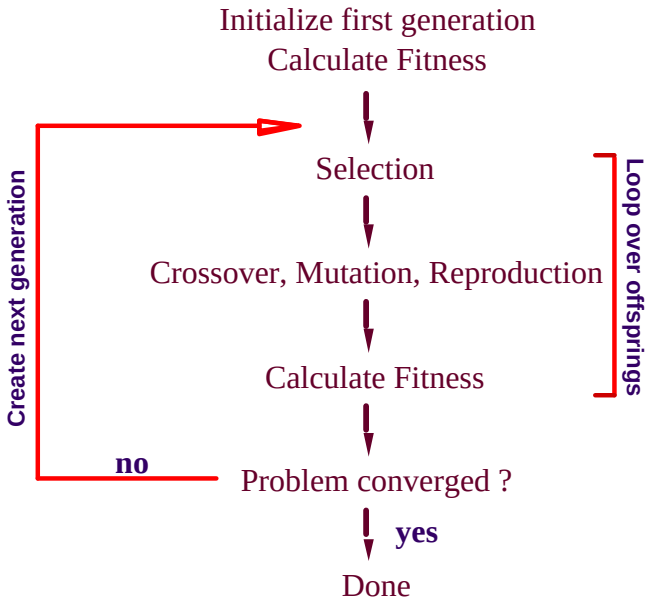

Figure 7. Solution procedure used.

The calculations were done on the CPLANT massively parallel computer at Sandia National Laboratories. Running on the parallel computer requires some slight modifications in the basic solution procedure to allow sharing of the "global-best" tree among processors. A processor is identified by its number, whose range is 0 to the number of processors allocated minus 1. In our implementation, process 0 is the master process which checks for convergence, collects the best trees from the other processors, determines which is the global-best tree, and handles sending this tree to the calculation processors. Each of the remaining processors run the same genetic program and run independently of the other processors.

At the end of each generation, each processor determines the best tree in its population and sends it to processor 0 where the globally best tree is determined. This send occur only if the local best tree has a larger fitness than the global-best it had received earlier. Processor 0 then determines if a new global-best tree is found and then broadcasts the new global-best tree to all processors. Accordingly, each processor decides whether the global-best tree will be employed in creating the next generation, using a rule that depends on the generation number. If a processor decides to use the global-best tree, the processor inserts that tree into the current generation to replace the tree that had the smallest fitness. In this way, the global-best tree is included in all of the genetic operations that produce the next generation.

The reason for not using the global-best tree all the time is to maintain diversity in the entire population. The convergence rate is proportional to a measure of the diversity. If all processors used the global-best tree in each generation, it would not be long before all of the trees would look much the same, and the rate of improvement would be reduced.

Figure 8 illustrates how the global-best tree is shared between the master processor and the calculation processors. The master processor receives the best trees from all of the calculation processors and then determines the current global-best tree. If the last time that a global-best tree was sent to all of the calculation processors is greater than the value of the parameter

SENDTIME, then the tree is sent. The tree will be sent to all of the calculation processors with PROBSND probability. Thus not all of the calculation processors will necessarily receive the global-best tree. The calculation processors will accept the global-best tree into its population of trees only if the current iteration is greater than the value USEGLOBALITER. These three parallel process variables were tested in this study.

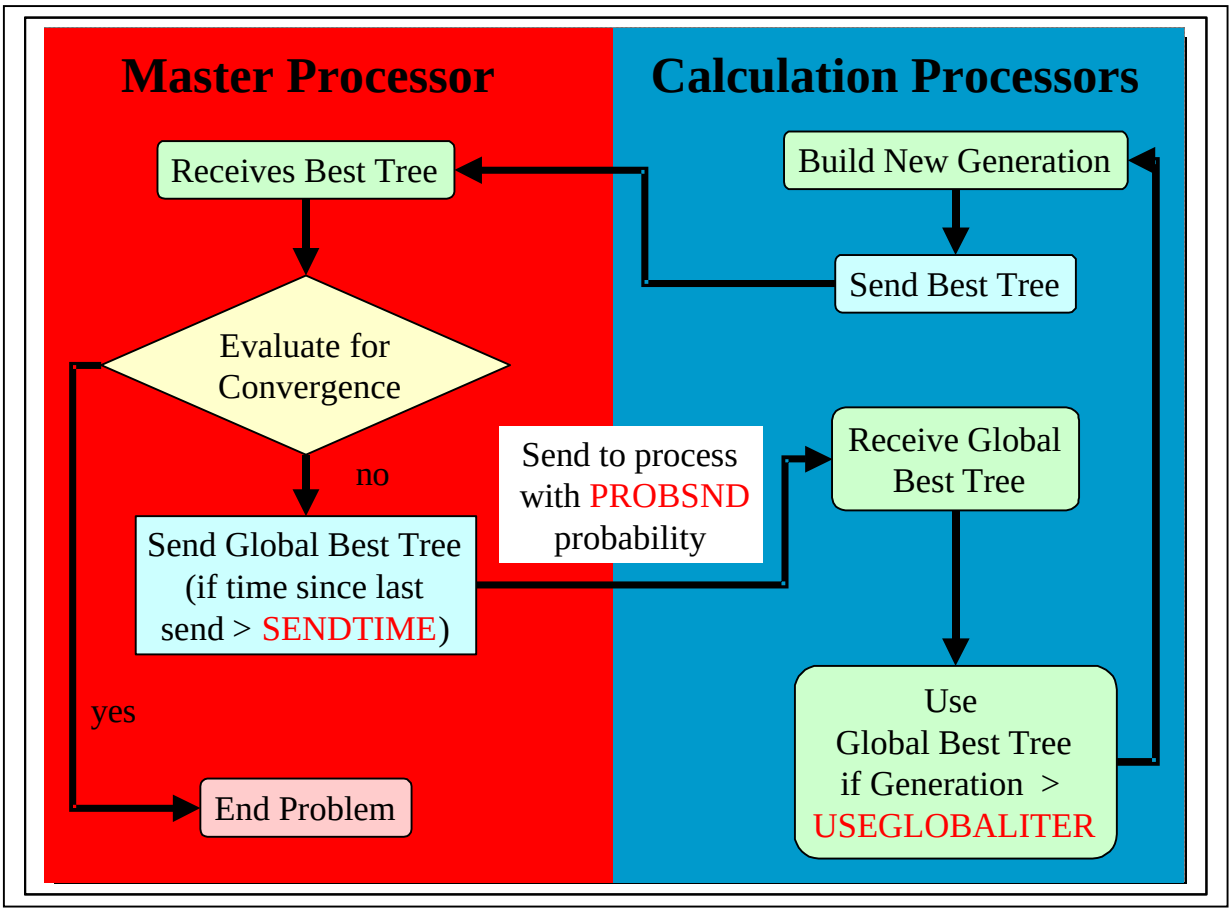

Figure 8 – Representation of Parallel Process Procedure.

The MAX Problem

Gathercole and Ross (1996) introduced the MAX problem to evaluate the standard GP crossover operator. The MAX problem uses a given function set and terminal set; and for a maximum tree depth, seeks the largest possible value. There are several advantages of the MAX problem for evaluating setup parameters for a GP method. The MAX problem has a known optimal value but can be a difficult problem for GP to solve. The results are known quickly and the solution space is easy to visualize. The problem can be varied with different function or terminal sets and changing the maximum tree depth allowed.

In this paper we used the function set {*,+} and one terminal of value 0.5. The root node counts as depth 0. For this version of the problem, in order to produce the largest value for tree depths of four or greater, the GP must build a tree where leaf terminals are combined with +

nodes to assemble subtrees with the value of 2.0. These must then be connected with either a + or * to give a higher level subtree value of 4.0. All nodes higher than this basic set must be *. The optimal base subtrees for this problem are shown in Figure 9.

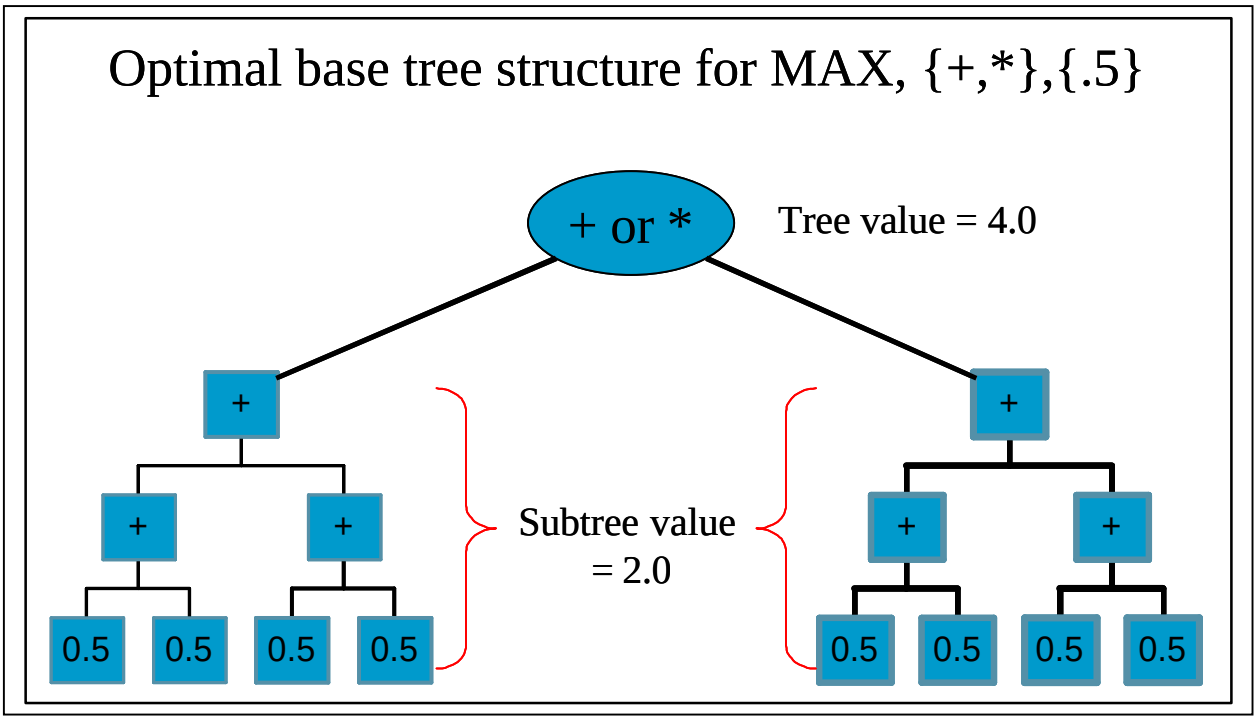

Figure 9 – Illustration of the base tree structure required to find the optimal value for MAX {+,*},{0.5}, Depth > 4.

A generic optimal tree of depth 4, is depicted in Figure 10. For a given depth D, the maximum possible tree value is $4^{2^{D-3}}$, where D >= 3. Thus the optimum value for the tree in Figure 10 where D is 4 (the root node counts as depth 0) is 16.

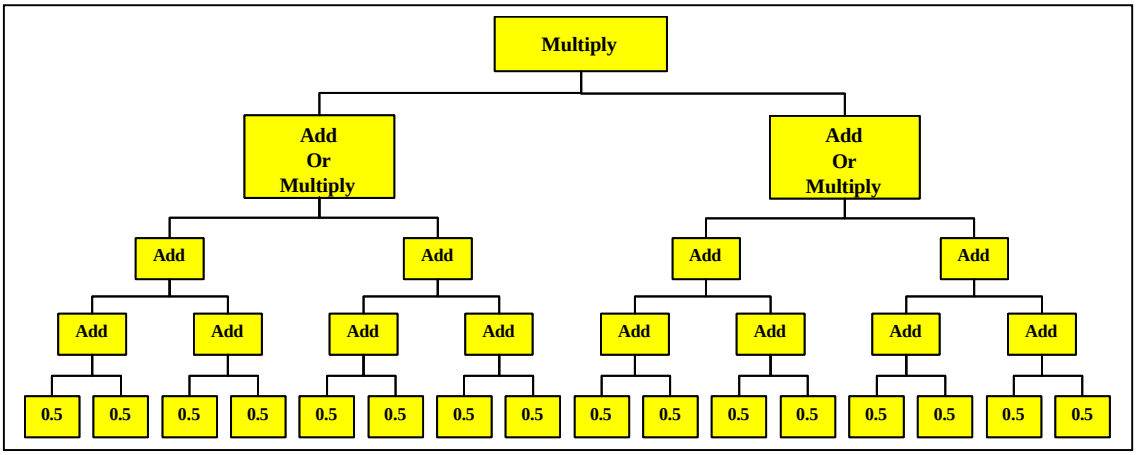

Figure 10 – An optimal tree for MAX depth 4, {+,*}, {0.5}

Parameter Evaluation Study

The parameters and the range of values that were evaluated are listed in Table 1. All parameters are described in the previous sections. The GP calculation parameters that were used in the study are listed in Table 2. The calculation parameters include variables like population size, maximum tree depth, and initial tree depth settings. The calcuation parameters were the same for all runs.

Table 1 – GP Test Parameters and Values

Parameter	Description	Range of Values
SELECTNUM	Number of individuals taking part in tournament selection	4 - 40
PROBCR	Crossover probability	.1 - 1.0
PROBMU	Mutation probability	.1 - 1.0
PROBLB	Limb mutation probability	.1 - 1.0
PROBND	Change node probability	.1 - 1.0
USEGLOBALITER	First iteration that the global best tree is accepted	0 - 450
PROBSND	Probability that the global best tree is sent to a process	.1 - 1.0
SENDTIME	Minimum time in between best tree send	0 - 135

Table 2 – Calculation Parameters

Parameter	Description	Value
MAXNODES	Maximum number of nodes	512
MAXTREEDEPTH	Maximum depth of final solution (root node = 0)	8
MAXINITDEPTH	Maximum depth of initial population of trees (root node = 0)	5
MININITDEPTH	Minimum depth of initial population of trees (root node = 0)	3
ANSWER	Solution to the MAX problem	18446744073709552000
POPSIZE	Population size	100
PROCESS	Number of nodes used on CPLANT	9

The computations were performed on Sandia's massively parallel CPLANT computer. The MAX problem was solved 500 times for each paramter value that was tested. Each new run was started with a new random number seed. The value used for all the variable parameter input values is shown in Table 3. These values were used for all of the problems except for the runs that tested PROBCR. For PROBCR the PROBMU value was set to 0, so the mutation operator was not used and the impact of crossover versus reproduction could be evaluated.

Table 3 – Standard Values for GP Method Variables Used in the Tests

Parameter	Standard Value
SELECTNUM	4
PROBCR	0.9
PROBMU	0.5 or * 0.1 for PROBCR
PROBLB	0.5
PROBND	0.2
USEGLOBALITER	50
PROBSND	0.5
SENDTIM	30

The problem was considered to be solved when the correct answer, ANSWER, for the MAX problem was discovered by one of the calculation processors and sent to the master processor. The problem was ended at this time and the solution time was recorded. The average solution time and standard deviation for the 500 runs are plotted for each input value in Figures 11 – 19.

Results

The following graphs display the average solution time and the standard deviation of the value for the 500 test runs. The first five graphs are parameters that control the genetic operations and the last three graphs are parameters that control the sharing of solutions across processors.

Genetic Operators

The results for the number of trees, SELECTNUM, used in the tournament selection process (Figure 11), show that the convergence time is little affected by the percentage of trees used in the tournament after about 10% of the population. The population size used in this test was 100. The selection procedure plays an important part in all of the genetic operations in this GP method.

The results of varying the probability of the mutation and crossover operations shows the interrelationship between the use of these two operations with the reproduction operation. Our results show that convergence time increases as the probability of mutation increases (Figure 12). When mutation becomes the primary genetic operation, i.e. when PROBMU is greater than .9, a solution for the MAX problem of depth 8 could not be found even after hours of searching. However, we also found that crossover without any mutation, i.e. when PROBMU = 0, can not solve the MAX problem of depth 8. Gathercole and Ross 1996 report similar results and the failure of the crossover operation alone in finding a solution for the MAX problem with depths greater than 6.

To test the relative role of reproduction and crossover, the mutation operator, PROBMU was set to a minimum value of 0.1, and PROBCR was varied from .2 – 1.0. The results of these tests are displayed in Figure 13. When PROBCR is low, the reproduction operator becomes the principal genetic operator in building the next generation. Reproduction does not create new trees but improves the fitness of the population through selection. Our results indicate that convergence time is slowed when reproduction is the predominant genetic operator. However,

when some reproduction is part of the genetic operations mix i.e., when PROBMU is .1 and PROBCR is .8 - .9, convergence improves by about 3% as compared to when no reproduction is used, i.e., when PROBMU is .1 and PROBCR is 1.0.

The results for the two different styles of mutation, limb or node, are shown in Figures 14 and 15. The results indicate that the style of mutation used has little effect on convergence time as long as limb mutation is not used exclusively (i.e. when PROBLB = 1.0). In addition, node mutation operation is best deployed on only a small number of the tree nodes (i.e. when PROBND is about .1).

Global-Best Tree Sharing

The parameters that control the sharing of the global-best tree across processors show little impact on convergence time (Figures 15 – 18). When the global best is shared at the earliest possible iteration (USEGLOBITER = 0), the convergence time is only slightly faster than sharing after 450 generations. When the probability that the calculation processors receive the global-best tree is low (PROBSND = .1) convergence time is only slightly improved over the possibility that all of the processors receive the global-best tree all of the time (PROBSND = 1). Convergence time is actually improved for smaller delays (SENDTIME < 15) for this problem, meaning that an earlier sharing of the global-best tree improves the efficiency of finding the correct solution. Gathercole and Ross, 1996, suggest that GP finds the MAX problem difficult because it quickly finds suboptimal solutions which contain + nodes near the root of the tree. Although sharing the global-best tree early in the problem reduces the diversity of the population, it helps to keep the calculation processors from concentrating on these suboptimal trees. This may not always be the best solution for other problems without many suboptimal solutions.

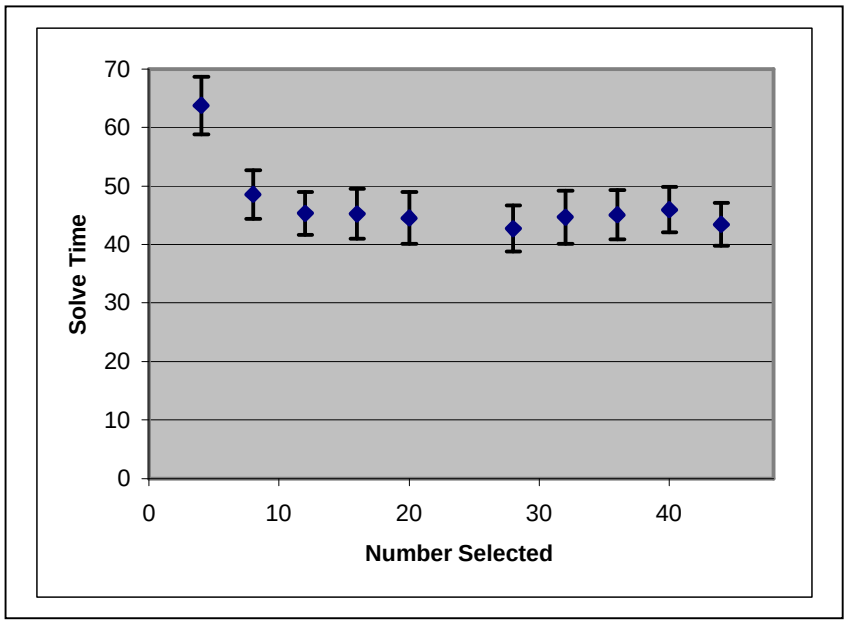

Figure 11 – Solution time for variation of the number of trees (SELECTNUM) chosen in the tournament selection process.

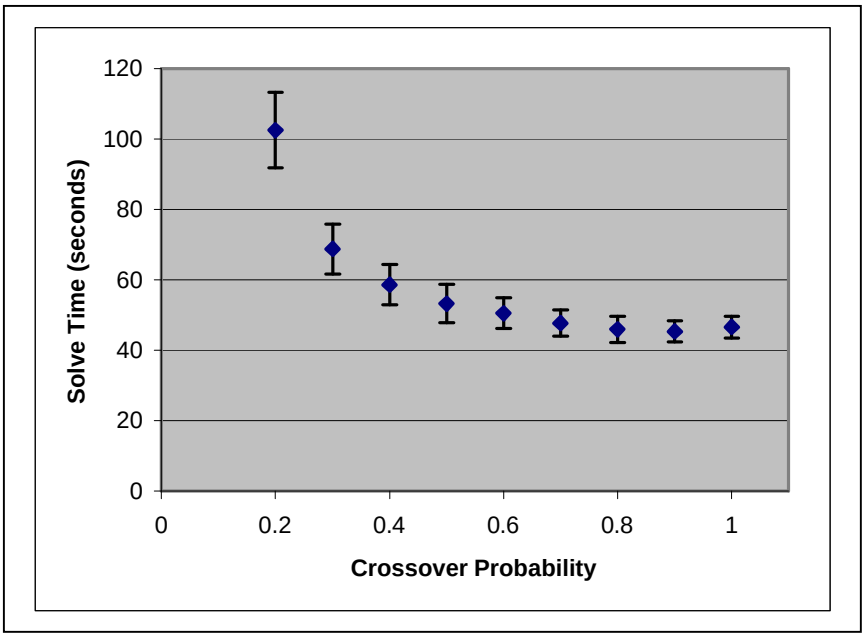

Figure 12- Solution Time for variation of mutation probability (PROBMU).

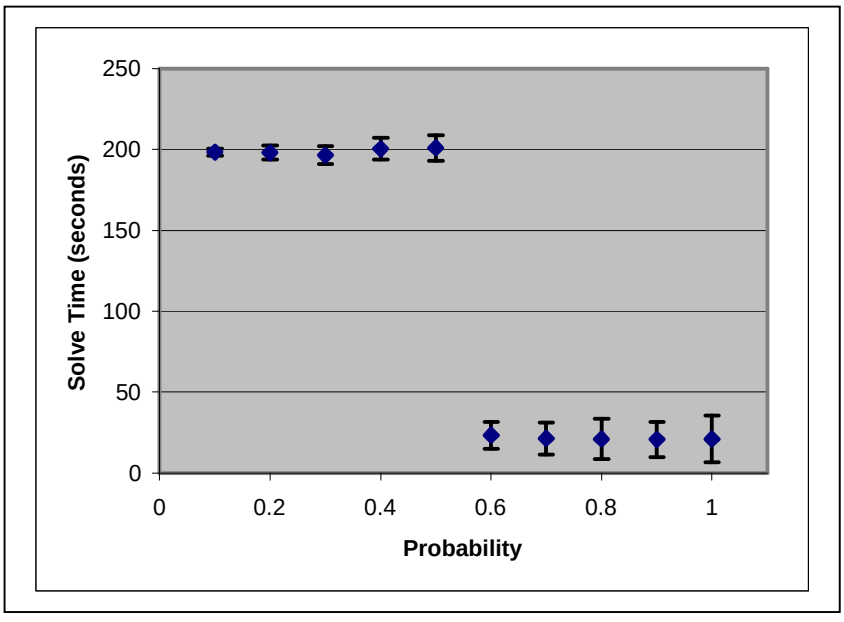

Figure 13– Solution Time for variation of crossover probability (PROBCR).

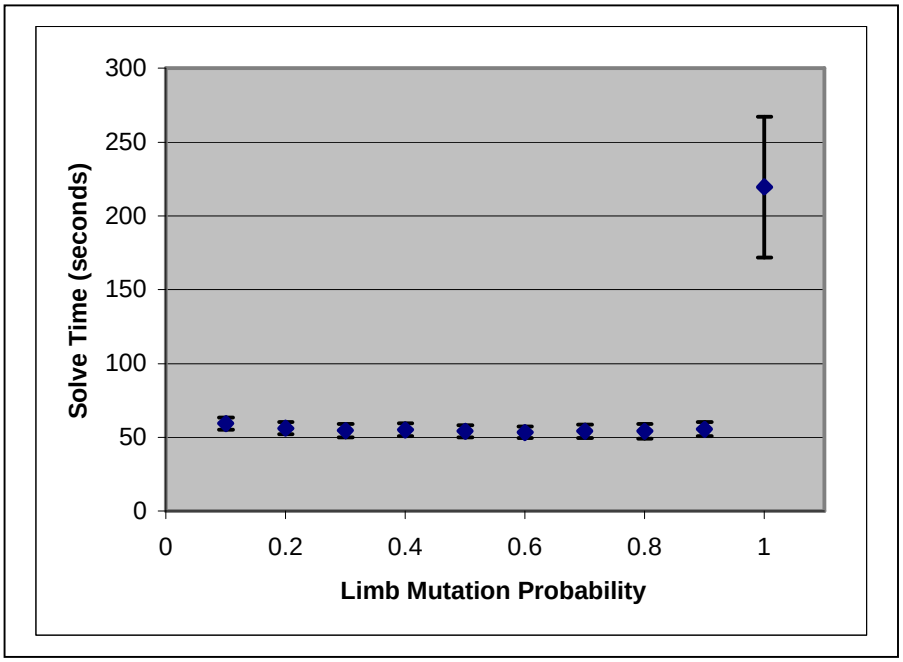

Figure 14 - Solution Time for variation of limb mutation probability (PROBLB).

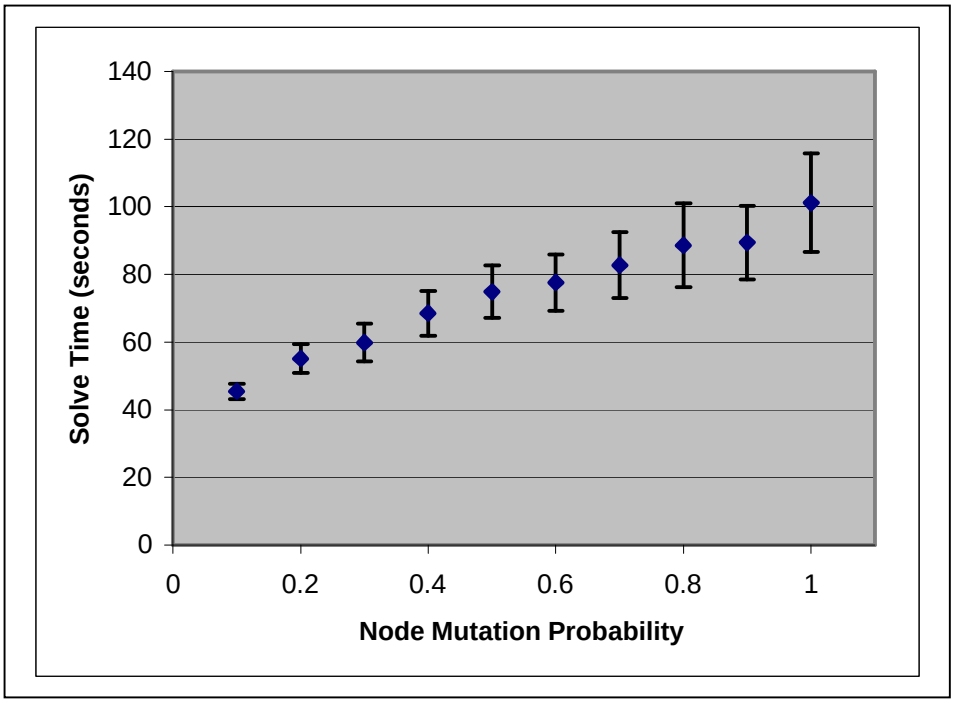

Figure 15 - Solution Time for variation of node mutation probability (PROBND).

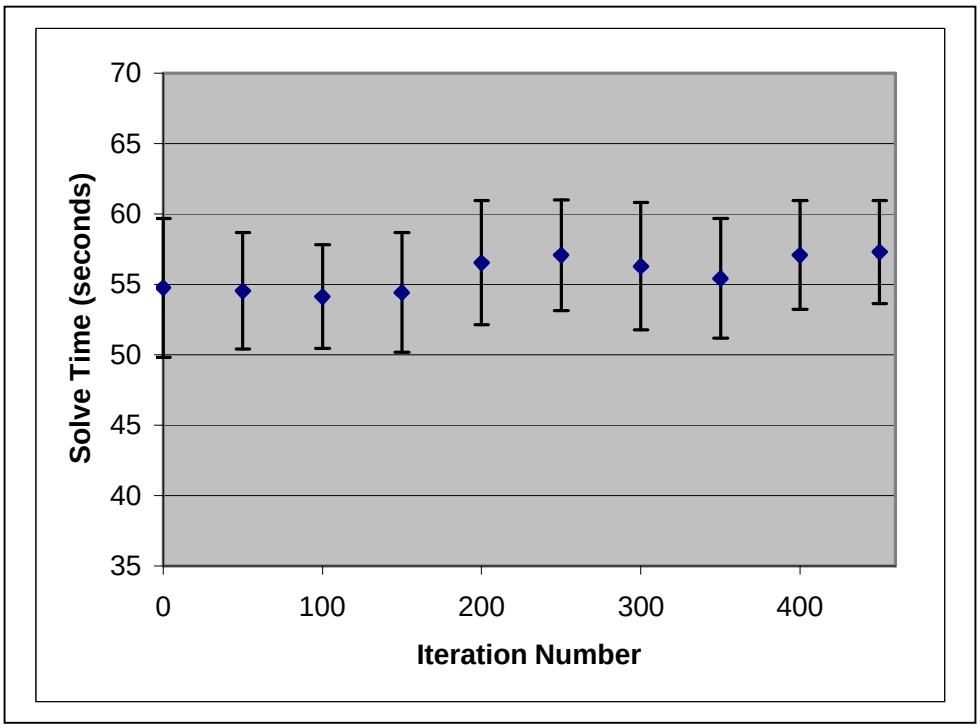

Figure 16 - Solution Time for variation of iteration that the global best tree is accepted by the process nodes (USEGLOBITER).

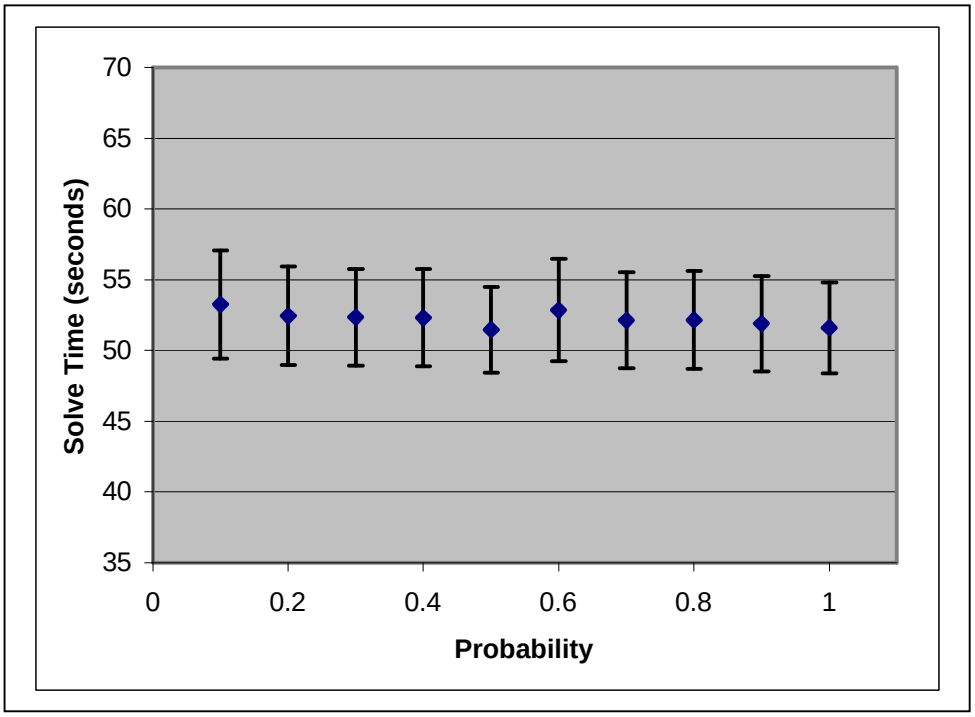

Figure 17 - Solution Time for variation of the probability that the global best tree is sent to the process nodes (PROBSND).

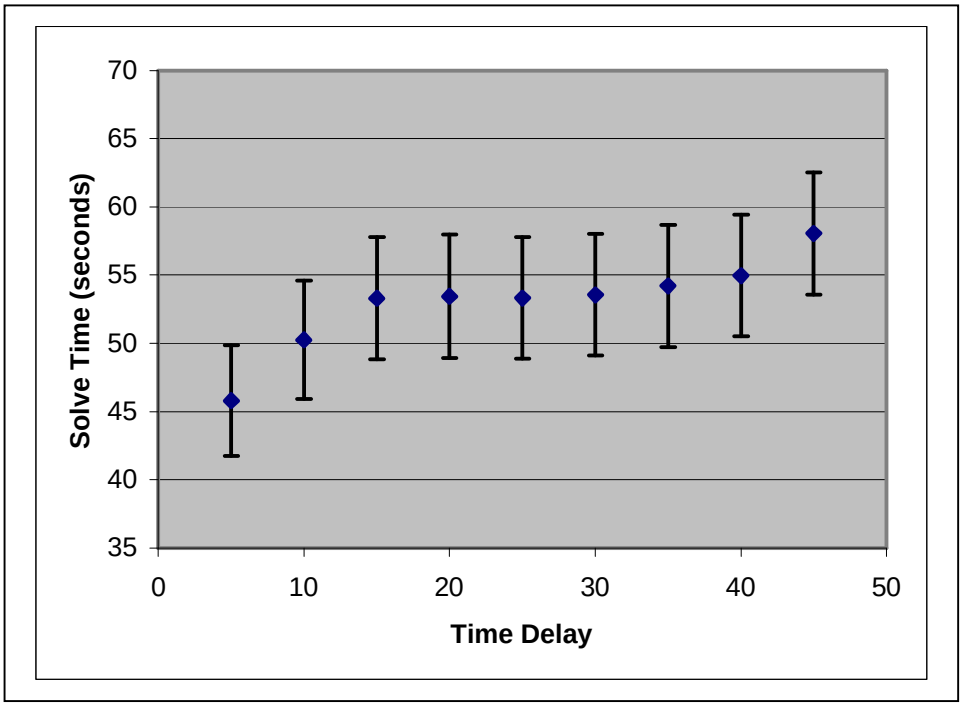

Figure 18 - Solution Time for variation of the wait time before the global best tree is sent to the process nodes (SENDTIME).

Conclusions

We have shown that the MAX problem is a useful and simple method to find the best settings for GP method parameters. The results of this study showed that steps taken to limit sharing of the global best tree had little effect on the results of the study. However, convergence time was strongly affected by the choice of genetic operation and the manner in which the genetic operations intermixed. Utilizing the MAX problem to tune the GP method parameters is a useful exercise and one that can help improve the efficiency of problem solving with GP.

References

Barton, Dianne C. *Using Genetic Programming to Develop Pricing Strategies for a Multi-Agent Economic Model.* SAND2002-3677. Albuquerque, NM: Sandia National Laboratories, 2002.

Goldberg, David E. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley, 1989.

Holland, John H. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.

Koza, John R. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press, 1992.

Pryor, Richard J. *Developing Robotic Behavior Using a Genetic Programming Model*. SAND98-0074. Albuquerque, NM: Sandia National Laboratories, 1998.

Pryor, Richard J., and Barton, Dianne. *Developing Maneuvering Behaviors for a Glider UAV Using a Genetic Programming Model* Albuquerque, NM: Sandia National Laboratories, (Draft report, September 2002).

Distribution

1	0321	W. J. Camp, 9200
1	0451	S. G. Varnado, 6500
1	0188	C. Meyers, 1030
1	0451	J. Nelson, 6515
40	1109	R. J. Pryor, 9212
40	1109	Dianne Barton, 6541
1	0129	N. Singer, 12620
1	9018	Central Technical Files, 8945-1
2	0899	Technical Library, 9616
1	0612	Review and Approval Desk, 9612, For DOE/OSTI